**Universidade Federal do Rio de Janeiro**

**Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais**

Patrick Müller de Andrade

# SEGURANÇA E MONITORAMENTO EM REDES PRIVADAS:

## Técnicas para Monitorar e Proteger

Rio de Janeiro

2013

**Patrick Müller de Andrade**

# MONITORAMENTO E SEGURANÇA EM REDES PRIVADAS:

## Técnicas para Monitorar e Proteger

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro – NCE/UFRJ

Orientador:

Moacyr Henrique Cruz de Azevedo, M.Sc., UFRJ, Brasil

Rio de Janeiro

2013

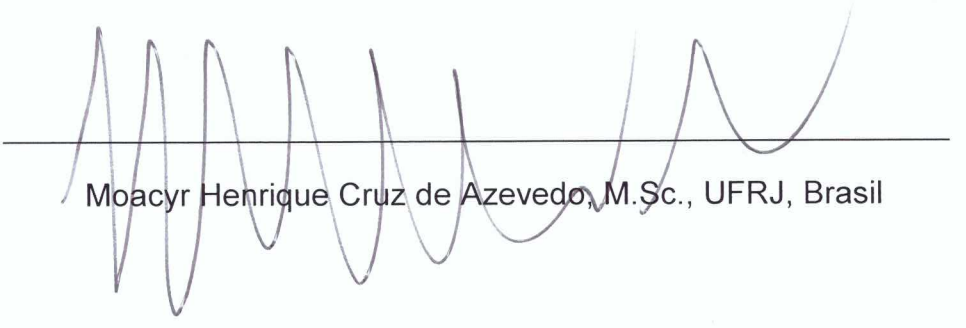
**Patrick Müller de Andrade**

# MONITORAMENTO E SEGURANÇA EM REDES PRIVADAS:

## Técnicas para Monitorar e Proteger

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro – NCE/UFRJ

Aprovada em março de 2013.



Moacyr Henrique Cruz de Azevedo, M.Sc., UFRJ, Brasil

Dedico este trabalho à todos que me incentivaram a me dedicar aos estudos e a nunca desanimar mesmo durante os momentos de dificuldades.

## AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dando força e foco para que eu conseguisse me concentrar nos estudos. Agradeço a minha família por me apoiar em todos os momentos, com um agradecimento especial à minha mãe e minha namorada. Agradeço aos meu amigos de trabalho e de curso por toda ajuda em momentos de dúvidas e dificuldades. Agradeço também à todos os professores pelas as excelentes aulas ministradas e um agradecimento especial ao meu orientador por se dedicar a me ajudar a concluir este trabalho.

## RESUMO

ANDRADE, Patrick Müller de. MONITORAMENTO E SEGURANÇA EM REDES PRIVADAS: Técnicas para Monitorar e Proteger. Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2013.

Este estudo aborda técnicas para a realização do trabalho de gerência de rede, como a configuração de serviços *Web* com o uso do *Apache*, de servidor de e-mail com o uso das aplicações *Postfix* e *Dovecot* e servidores de banco de dados com *Mysql.* As configurações sugeridas buscam obter o máximo de segurança. Além disso, foi realizado o monitoramento desses aplicativos, servidores e equipamentos ativos da rede com o uso de softwares como o *Nagios*, *Cacti* e PRTG.

## ABSTRACT

ANDRADE, Patrick Müller de. MONITORAMENTO E SEGURANÇA EM REDES PRIVADAS: Técnicas para Monitorar e Proteger. Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2013.

This study discusses network management techniques, such as Apache web services settings, e-mail server using Postfix and Dovecot applications, and Mysql data bases servers. The suggested settings try to get the maximum security. In addition, the monitoring of the applications, servers and equipaments using softwares such as *Nagios*, *Cacti* and PRTG was done.

## LISTA DE FIGURAS


Página

Figura 1 - Topologia Simplificada 15
Figura 2 - Conhecimento Técnico do Invasor x Sofisticação Ataque 16
Figura 3 - Ilustração do posicionamento de um *firewall* 17
Figura 4 - Ocorrência de violação do uso do *Switch* 22
Figura 5 - Configuração para tunelamento sobre SSH 23
Figura 6 - Acesso remoto por Remote desktop 24
Figura 7 - Estabeleciamento SSH prompt Windows 24
Figura 8 - Conexão SSH e Remote Desktop porta 3390 do endereço de localhost 25
Figura 9 - Estabelecimento SSH e *Remote Desktop* servidor. 25
Figura 10 - *Scanning* não encontra a porta aberta. 25
Figura 11 - Diferença em uma tentativa de acesso a um diretório não existente 29
Figura 12 - Diferença em um scanning 30
Figura 13 - *Software* MIB *Browser* 45
Figura 14 - Informação de um Bloqueio de *spam* gerado pelo *logcheck* 46
Figura 15 - Informação de vulnerabilidade gerada pelo *Portaudit* 47
Figura 16 - Gerenciamento do *status* do serviço pelo Nagios 48
Figura 17 - Monitoramento de um servidor com o Cacti 49
Figura 18 - Monitoramento de um *switch* com o *Cacti* 49
Figura 19 - Monitoramento dos eventos do sistema de uma máquina com o PRTG. 50
Figura 20 - Monitoramento dos eventos das aplicações de uma máquina com o PRTG 51
Figura 21 - Trechos monitorados com o PRTG. 52
Figura 22 - Topologia da rede monitorada 52
Figura 23 - Gráfico da variação do jitter dos trechos monitorados com o PRTG. 53

## LISTA DE QUADROS


Página

Quadro 1 - Regras tipo *stateless* usando *ipfw* 18
Quadro 2 - Regras tipo statefull usando ipfw 18
Quadro 3 - Regras dinâmicas geradas pelas regras com opção *keep-state* 18
Quadro 4 - Configuração Port Security 22
Quadro 5 - Regra para acesso remoto por tunelamento SSH 23
Quadro 6 - Configuração de permissão de diretório no *Apache* 30
Quadro 7 - Configuração de ACL no DNS 34
Quadro 8 - Exemplo de configuração de *view* no DNS 34
Quadro 9 - Configurações do Postfix para acesso externo seguro 36
Quadro 10 - Configuração do Dovecot para acesso externo seguro 37
Quadro 11 - Configuração do *Postfix* para bloqueio de *spans* 1 38
Quadro 12 - Configuração do *Postfix* para bloqueio de *spans* 2 38
Quadro 13 - Configuração do *Postfix* para bloqueio de *spans* 3 38
Quadro 14 - Configuração do *Postfix* para bloqueio de *spans* 4 39
Quadro 15 - Configuração para checagens de portas através do *Nagios* 48

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| ACK | Acknowledged |
| AES | Advanced Encryption Standard |
| DHCP | Dynamic Host Configuration Protocol |
| DMZ | Demilitarized Zone |
| DNS | Domain Name System |
| DNSSEC | Domain Name System Security Extensions |
| DOS | Denial of Service |
| DDOS | Distribut Denial of Service |
| FIN | Finished |
| HTTP | HyperText Transfer Protocol |
| HTTPS | HyperText Transfer Protocol Secure |
| ICMP | Internet Control Message Protocol |
| IDS | Intrusion Detection System |
| IMAP | Internet Message Access Protocol |
| IP | Internet Protocol |
| PHP | Hypertext Preprocessor |
| POP3 | Post Office Protocol |
| VLAN | Virtual Local Area Network |
| VPN | Virtual Private Network |
| SASL | Simple Authentication and Security Layer |
| SNMP | Simple Network Management Protocol |
| SSH | Secure Shell |
| SYN | Synchronize |
| TCP | Transmission Control Protocol |
| TLS | Transport Layer Security |
| UDP | User Datagram Protocol |
| WEP | Wireless Equivalente Privacy |
| WPA2-PSK | Wi-Fi Protected Access 2 Pre-Shared Key |

SUMÁRIO


1 **INTRODUÇÃO** 11
2 **TOPOLOGIA DA REDE** 15
3 **SEGURANÇA** 15
3.1 FIREWALL 17
3.1.1 **Firewall Filtro de Pacotes** 18
3.1.2 **Firewall Gateway de Aplicação** 19
3.2 SEGURANÇA INTERNA 20
3.2.1 **Segurança em VLANs** 20
3.2.2 **Acesso Remoto** 22
3.2.3 **Segurança em Redes sem Fio** 25
3.3 SEGURANÇA DOS SERVIÇOS E SISTEMAS 26
3.3.1 **Segurança em Servidores Web** 28
3.3.1.1 Segurança no Apache 29
3.3.1.2 Segurança do PHP 31
3.3.2 **Segurança do Servidor DNS** 32
3.3.3 **Segurança no Servidor de E-mail** 35
3.3.4 **Segurança dos Servidores** 39
4 **MONITORAMENTO** 43
4.1 MONITORAMENTO DE SEGURANÇA 43
4.1.1 **IDS** 43
4.1.2 **Alertas via SNMP** 44
4.1.3 **Monitoramento de Logs e Verificação de Vulnerabilidades** 45
4.2 MONITORAMENTO DA REDE 47
4.2.1 **Monitoramento dos Serviços** 47
4.2.2 **Monitoramento de Sistemas e Ativos** 48
4.2.3 **Monitoramento do Estado da Rede** 52
5 **LICENCIAMENTO DOS SOFTWARES** 54
6 **CONCLUSÃO** 56
7 **REFERÊNCIAS** 57

# 1 **INTRODUÇÃO**

Esse projeto será desenvolvido devido à necessidade de monitoramento e uma melhora da segurança da rede nos laboratórios já existentes e principalmente em um novo laboratório que será inaugurado, todos sobre a responsabilidade da mesma gerência de TI.

Esse projeto visa documentar para então padronizar a implementação dos serviços de rede necessários aos usuários de uma rede corporativa como: acesso à rede, e-mail coorporativo, armazenamento seguro de dados dos projetos, acesso remoto, acesso à internet, visando o máximo de segurança e monitoramento dos serviços oferecidos a esses usuários. A seguir serão descritas algumas das diretrizes do funcionamento dessa estrutura.

O acesso à rede só poderá ser feito por máquinas cadastradas previamente, seja na rede cabeada ou *Wireless*. Os grupos serão divididos em *Virtual Local Area Network* (VLAN). Como não há necessidade de mobilidade das máquinas cadastradas na rede cabeada, haverá um controle de acesso à rede por *Port Security*, para assim ter um controle sobre alguma tentativa de uso da rede por alguém não autorizado. Já o acesso à rede *Wireless* será feito por meio de autenticação em um servidor *Radius*.

O e-*mail* coorporativo poderá ser acessado por *Web-mail*, *smartphone* ou aplicativos de gerenciamento, tanto da rede interna como da externa. Para garantir a segurança desse serviço serão feitas uma série de configurações de segurança nos servidores que serão usados para disponibilizar esse serviço.

Para atender a necessidade de alguns grupos de acessar remotamente suas máquinas, poderá ser usada uma das soluções: *Virtual Private Network* (VPN), técnica que emula uma rede privada em uma rede pública, ou tunelamento por *Secure Shell*

(SSH), que é um protocolo da camada de apresentação cuja sua principal função é prover comunicação criptografada, sendo assim ideal para comunicação remota.

O acesso à internet dos usuários será controlado por *Proxy*, facilitando o gerenciamento do que é ou não permitido acessar, quem pode usar a internet, e principalmente facilitar a implementação de regras para aumentar a segurança do uso da internet e a adequação de regras de *firewall*.

Já para atender a necessidade de visitantes de acesso à internet, será implementada uma rede totalmente independente, sem nenhuma comunicação à rede do laboratório, somente para acesso à internet. Nessa rede a autenticação será feita por meio de senha usando método de autenticação *WPA2-PSK*, onde é somente necessária uma senha para ter-se acesso à rede.

As necessidades mencionadas requerem uma série de serviços e servidores disponíveis. Para manter toda essa infraestrutura funcional há necessidade de monitoramento dos servidores e ativos da rede, além de uma grande atenção com a segurança desses dispositivos.

Então para fazer o monitoramento tanto da disponibilidade quanto da segurança de servidores, serviços e ativos de redes serão implementados serviços como: *Simple Network Management Protocol* (SNMP), que são mensagens trocadas entre o gerente e o agente a fim de obter alguma informação necessária.

Na parte de segurança serão usados *firewall*, solução que pode envolver vários equipamentos cuja função é impedir a entrada e saída de comunicação indesejada; *Intrusion Detection System* (IDS), que é uma solução de segurança passiva, sua função é verificar os pacotes que estão trafegando pela rede a fim de identificar uma possível

falha de segurança; e *Hardening* de serviços e sistemas, que são técnicas para tornar o serviço ou servidor mais robusto.

Com o uso de *softwares* como *Nagios* e *Cacti* que trabalham com informações SNMP, serão monitorados os servidores e equipamentos de rede. O *Nagios* será usado para monitorar o estado dos serviços disponibilizados. Já o *Cacti* será usado para monitorar o tráfego e processamento dos *switches,* além de outros aspectos dos servidores como: trafego de rede, processamento, consumo de memória e disco.

Com o uso do *Snort*, que é um software de IDS que pode trabalhar por meio de assinaturas, comportamento ou ambos, será feito o monitoramento de tentativas de acesso não permitido, principalmente de caráter interno. Será usado para monitorar máquinas críticas, como as dos setores financeiro e administrativo.

Com técnicas de *Hardening,* que serão rechecadas periodicamente, tem-se como objetivo identificar e corrigir possíveis falhas de seguranças dos serviços disponíveis, fortalecer servidores contra ataques e os tornar obscuros visando ocultar o máximo de informações sobre os serviços ativos.

O trabalho será estruturado da seguinte forma: todas as técnicas citadas serão implementadas em uma rede de teste, formada por uma máquina que fará a função de *gateway*, com *firewall, proxy* e DNS além de ser responsável por fazer o roteamento entre as VLANs. Uma segunda máquina fará as funções de hospedagem de *sites* e servidor de e-*mail* e uma terceira máquina terá os aplicativos de monitoramento mencionados.

## 2 **TOPOLOGIA DA REDE**

A topologia da rede onde as técnicas serão mostradas é descrita a seguir. Terá um *gateway* único, que será uma máquina com sistema operacional FreeBSD que terá as funções de roteamento entre redes e VLANs. A rede será dividida em duas partes: uma rede com IPs válidos, *Demilitarized Zone* (DMZ), onde ficarão todos os servidores que precisarão ser acessados externamente, e a rede interna formada por VLANs com IPs privados. Essa solução será adotada para aumentar a segurança da rede interna, pois caso algum destes servidores sejam comprometidos os danos causados a rede interna será o menor possível.

As outras funções desse *gateway* são *firewall* da rede, servidor de DNS e DHCP, sendo assim todos os serviços básicos da rede serão fornecidos por essa máquina. Na rede DMZ também estarão às máquinas responsáveis pelos serviços de *e-mail*, *Web Service* e acesso remoto. A máquina responsável pelo banco de dados estará em uma VLAN e as outras máquinas que serão usadas para fazer o monitoramento estarão na VLAN de administração. A figura 1 ilustra de forma resumida essa topologia.

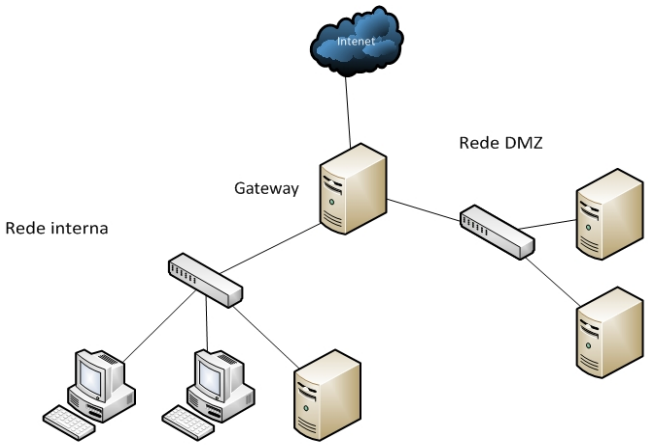



Figura 1 – Topologia Simplificada

# 3 **SEGURANÇA**

Com o crescimento do uso da *internet* o assunto segurança em redes se tornou muito importante independente do tipo de negócio da empresa, afinal nenhum gerente ou dono de empresa quer que seus segredos de negócios, informações de clientes ou de projetos sejam acessados por pessoas não autorizadas. Um estudo datado de 2002 mostra que com o passar do tempo cada vez mais o assunto segurança em redes deve ser tratado com mais atenção. Esse estudo mostra a evolução do poder destrutivo de ataques versos a necessidade de conhecimento do atacante (Figura 2). Como atualmente a maioria dos ataques são feitos por *scripts*, que são códigos de programação para efetuar um função, qualquer pessoa má intencionada pode fazer um ataque considerado de alto nível sem muitos conhecimentos de rede e seus protocolos.

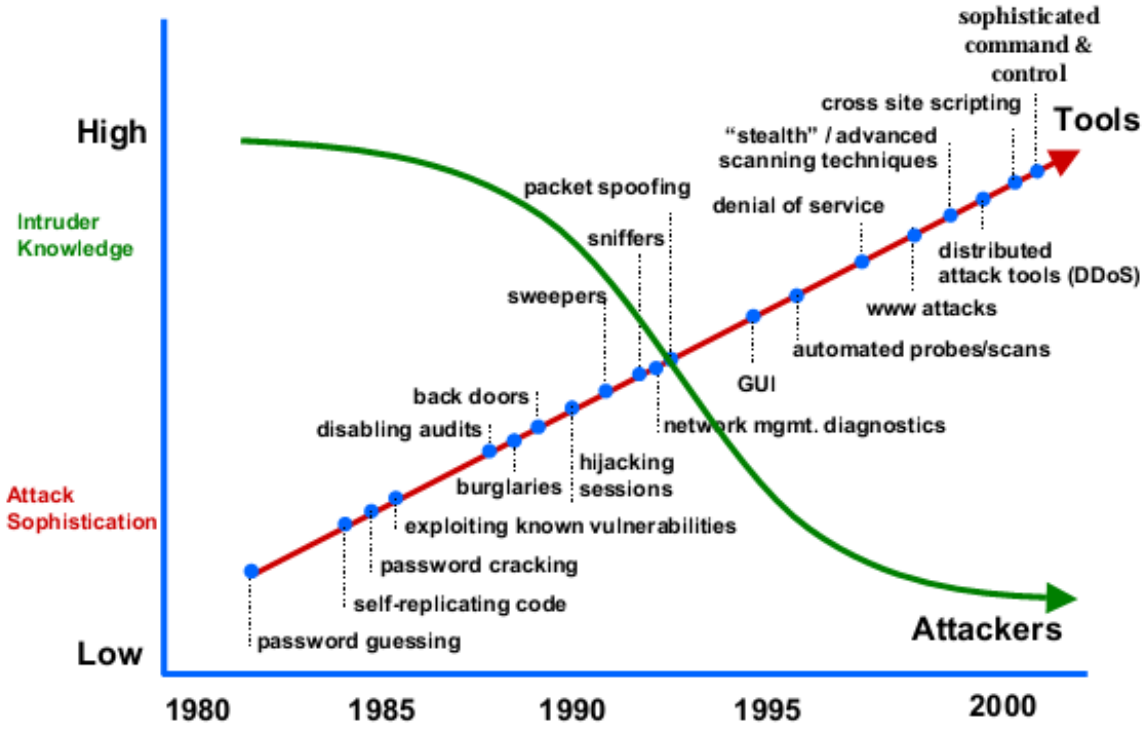



Figura 2 - Conhecimento Técnico do Invasor x Sofisticação Ataque [3]

Na figura 2 a linha *Tools* representa o nível de conhecimento do atacante, já a linha A*ttackers* mostra o poder destrutivo do ataque. Essa figura mostra com muita clareza o quão importante é se tratar com atenção a questão da segurança de uma rede. Por isso quando trata-se deste assunto deve-se assumir como fundamentais as três dimensões: confidencialidade, que trata da manutenção do sigilo ou privacidade da informação; integridade, que trata da manutenção dos dados; e disponibilidade, que trata da acessibilidade da informação para o seu uso de forma contínua e ininterrupta. Nesse capítulo serão abordadas algumas técnicas de segurança necessárias em uma rede.

## 3.1 FIREWALL

Começando com o básico: controle de acesso feito por *firewall*. Um *firewall* é uma solução de *hardware* e *software* que terá a função de controlar o que pode entrar e sair de uma rede, sendo denominada uma solução de perímetro (figura 3).

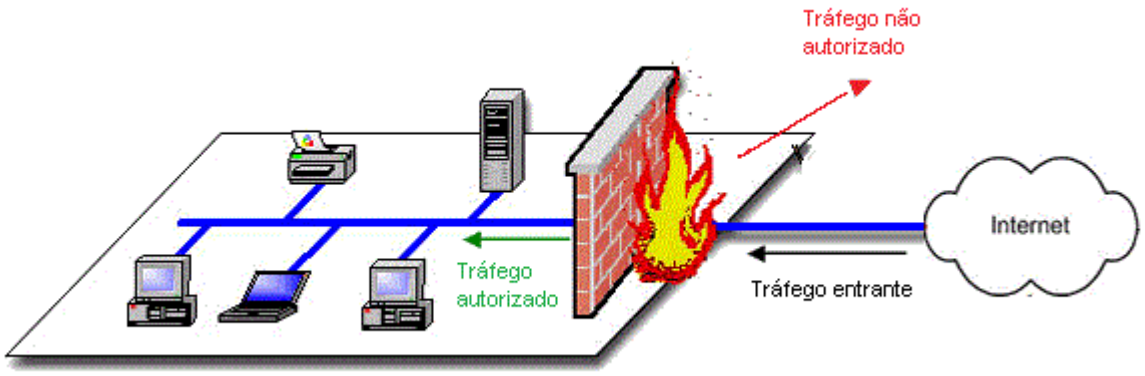



Figura 3 - Ilustração do posicionamento de um *firewall*

Existem dois tipos de *firewall*: filtro de pacotes e *gateway* de aplicação.

### 3.1.1 ***Firewall*** **Filtro de Pacotes**

Um *firewall* de filtro de pacotes pode trabalhar em dois modos, *stateless* e/ou *statefull*. Um *firewall* trabalhando em modo *stateless*, não guarda o estado das conexões, exemplo de regras desse tipo são:

Quadro 1 – Regras tipo *stateless* usando *ipfw*

| Ação | protocolo | origem | para | destino | opções |
|---|---|---|---|---|---|
| Allow | udp | from ${net_n}:${mask_n} | to | Any | 53 |
| Deny | all | from any | to | Any | |

A primeira regra implementa uma permissão de conexão via protocolo UDP de qualquer IP de uma determinada rede e sua máscara de rede, para qualquer IP na porta 53. Essa regra é usada para consultas de DNS. A segunda regra implementa um bloqueio de qualquer comunicação, usada para fechar um *firewall.*

Já o modo *statefull,* além de guardar o estado das conexões, ainda tem a capacidade de criar regras dinâmicas, exemplo de regras:

Quadro 2 – Regras tipo *statefull* usando *ipfw*

| ação | protocolo | Origem | para | destino | opções |
|---|---|---|---|---|---|
| allow | Tcp | from any | to | any | 22 keep-state |
| allow | Tcp | from any | to | any | 443 keep-state |

Ambas as regras são de permissão de conexão de qualquer IP de origem para qualquer IP de destino nas portas 22 e 443 usando o protocolo TCP. Respectivamente significam permissão de acesso a SSH e HTTPS.

Quadro 3 – Regras dinâmicas geradas pelas regras com opção *keep-state*

| timeout | estado | protocolo | origem | porta | destino | porta |
|---|---|---|---|---|---|---|
| 12s | STATE | tcp | 146.164.111.111 | 3482 | 146.146.222.222 | 22 |
| 8s | STATE | tcp | 146.164.111.111 | 58872 | 50.22.231.49 | 443 |
| 18s | STATE | tcp | 146.164.111.111 | 56770 | 74.125.234.33 | 443 |
| 8s | STATE | tcp | 192.168.254.35 | 58872 | 50.22.231.49 | 443 |
| 18s | STATE | tcp | 192.168.254.36 | 56770 | 74.125.234.33 | 443 |

As regras mostradas no quadro 3 foram criadas dinamicamente pela utilização da opção *keep-state* descritas no quadro 2. Essas regras são criadas devido à função de armazenamento do estado das conexões implementada por um *firewall statefull.*

Para um *firewall* de filtro de pacotes determinar se um pacote é ou não válido, serão verificados alguns campos dos cabeçalhos desse pacote para então serem aplicadas suas regras. Os campos verificados nesse pacote são: endereços IP de origem e destino, portas TCP ou UDP de destino e origem, e os bits SYN, ACK e FIN do cabeçalho TCP. A ação aplicada após essas analises pode ser de permissão ou negação, onde a permissão significa a aceitação desse pacote e a negação significa o descarte.

Um filtro de pacotes *stateless* funciona de forma sequêncial, quer dizer que será feita uma verificação linha a linha do *script* de *firewall* a procura de uma ação a ser tomada, seja de permissão ou negação. Sendo assim é muito importante verificar com cuidado a sequência das regras, pois o posicionamento errado de uma regra pode parar um tráfego válido ou até criar uma falha de segurança.

Já o filtro de pacote *statefull* não trata pacote a pacote, ele trata os fluxos das conexões TCP ou UDP. Esse controle é realizado com base no número de sequência dos pacotes, para assim determinar se um tráfego faz ou não parte de uma conexão permitida ou negada.

### 3.1.2 ***Firewall Gateway*** **de Aplicação**

Só um filtro de pacote não traz toda a segurança necessária. Para complementar a solução de *firewall* é importante ter um *gateway* de aplicação, por exemplo, um *proxy*. Esse tipo de *firewall* faz um controle mais refinado, além de poder trabalhar em associação com outros *softwares*. Um exemplo de *software* que pode ser associado ao

*proxy* é o *DansGuardian* [5], que é um aplicativo com a função de filtragem de conteúdo. Essa filtragem é realizada através de regras adaptativas que avaliam se o conteúdo de um *site* é ou não próprio.

Outro *software* interessante para se agregar ao *proxy* é um antivírus como o *ClamAV*, que com um método de funcionamento parecido com o *DansGuardian* avaliará os arquivos requisitados pelos usuários para permitir ou não o seu *download*. Com esse conjunto de aplicativos é possível fornecer um acesso à *internet* mais seguro.

Mas ainda que o *firewall* esteja bem configurado ele estará suscetível a falhas inerentes à arquitetura TCP-IP, como por exemplo, o *spoofing*. Como o protocolo IP não tem nenhum tipo de autenticação ele pode ser forjado, de forma que um intruso possa adulterar seu IP para tentar burlar o *firewall.* Outro ponto fraco de um *firewall* é não ter nenhuma efetividade contra ataques de *Denial of Service* (DOS) e sua variação mais poderosa *Distribut Denial of Service* (DDOS), cuja idéia é derrubar um sistema através de múltiplas requisições. Geralmente essa técnica é usada em associação com *spoofing* ou com uso de *botnets,* as conhecidas redes zumbi.

Apesar de possuir falhas, a solução de *firewall* usando filtro de pacotes *stateless* e *statefull* mais *gateway* de aplicação traz um bom nível de segurança para uma rede.

## 3.2 SEGURANÇA INTERNA

### 3.2.1 **Segurança em VLANs**

Uma vez feita uma proposta de solução de segurança contra ataques externos, precisa-se fazer o mesmo para a rede interna. Como foi visto anteriormente, a rede interna será dividida em sub-redes, formadas por VLANs. Essa solução será adotada principalmente pelo fato de alguns grupos de usuários não terem suas máquinas

administradas pela equipe de suporte e rede do laboratório, além de muitas destas máquinas serem computadores portáteis pessoais. Sendo assim é necessário haver um controle interno, principalmente da comunicação entre as VLANs.

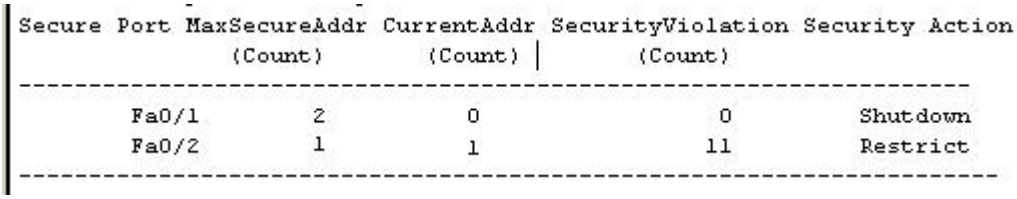
A comunicação entre as VLANs será controlada pelo *gateway*, que é a máquina responsável pelo tráfego entre a rede interna e a rede externa, pois essa máquina possui um *firewall.* Assim tem-se a possibilidade da criação de regras para controlar a comunicação entre as VLANs.

Além do controle por VLAN, todas as máquinas que acessam a rede deverão ser previamente cadastradas, pois os IPs fornecidos para as máquinas serão fixos, ou seja, toda vez que uma determinada máquina se conectar a rede ela receberá o mesmo endereço. Além disso, nos *switches,* equipamento responsável pela comunicação entre as máquinas em uma mesma VLAN, será implementada a técnica de segurança chamada de *Port Security* [16]*.* Essa técnica tem a função de verificar se o computador que está tentado se conectar a uma determinada porta do *switch* tem essa autorização. Esse controle é feito pelo endereço *MAC*, que é o endereço da interface de rede de uma máquina. Então, quando uma máquina se conectar a uma porta do *switch,* ele aprenderá esse endereço e somente será permitido a esse *MAC address* se conectar novamente a essa porta. Caso uma máquina não autorizada seja conectada a essa porta do *switch*, ela ficará bloqueada até a máquina autorizada se conectar novamente, um exemplo dessa configuração é dada no quadro 4.

Quadro 4 – Configuração *Port Security*

| Commando | Ação |
|---|---|
| switchport mode access | Mudança de modo de funcionamento da interface |
| switchport port-security | Habilita a função de *Port Security* na interface |
| switchport port-security maximum 1 | Limita a quantidade de MACs que poderá se conectar a essa porta em apenas um. |
| switchport port-security mac-address sticky | Determina que o MAC seja aprendido pelo *switch,* o MAC aprendido será o primeiro a se conectar à porta. |
| switchport port-security violation restrict | Restringe o uso da porta somente ao MAC previamente aprendido |

A figura 4 exemplifica uma possível violação desse mecanismo de segurança, ao acessar o *switch* e realizar uma consulta será obtida a informação mostrada.



| Secure Port | MaxSecureAddr (Count) | CurrentAddr (Count) | SecurityViolation (Count) | Security Action |
|---|---|---|---|---|
| Fa0/1 | 2 | 0 | 0 | Shutdown |
| Fa0/2 | 1 | 1 | 11 | Restrict |

Figura 4 – Ocorrência de violação do uso do *switch*

É claro que mesmo com essas técnicas um usuário mal intencionado, com conhecimento de como alterar o *MAC address* de um computador, poderá acessar a rede, desde que ele conheça o *MAC address* autorizado.

### 3.2.2 **Acesso Remoto**

Uma necessidade dos usuários que vem aumentando é a questão do acesso remoto as suas máquinas de trabalho. A solução proposta para atender a essa demanda é o acesso ser feito por meio de tunelamento via SSH. Para isso é preciso haver uma máquina na rede DMZ que possa se comunicar com as máquinas da rede interna. Essa comunicação será feita por *Remote Desktop*, no entanto para essa comunicação ser possível é preciso ter uma regra no *firewall* que permita o acesso

somente às máquinas autorizadas, um *software* que permita criar tuneis via SSH, por exemplo o *Secure Shell* SSH.

A seguir serão abordados os passos dessa configuração. O exemplo da regra necessária no *firewall* é vista no quadro 5.

Quadro 5 – Regra para acesso remoto por tunelamento SSH

| Ação | protocolo | origem | para | destino | Opções |
|---|---|---|---|---|---|
| Allow | tcp | from 146.164... | to | 192.168.254.27 | 3389 keep-state |

Agora é preciso realizar as configurações no software que criará o túnel. O exemplo será feito para uma comunicação por *Remote Desktop*. A figura 5 mostra os passos necessários para essa configuração.

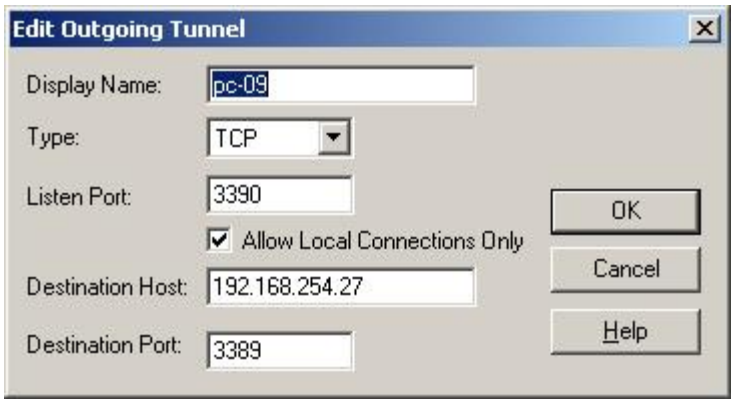



Figura 5 – configuração para tunelamento sobre SSH.

Na configuração o *Listen Port* é a porta local que a máquina que realizará a comunicação usará para a conexão. *Type* é o protocolo que será usado nessa comunicação. *Destination Host* é o IP da máquina a qual se quer acessar, seja ele um IP válido ou no caso do exemplo um IP inválido. A *Destination Port* é a porta a qual a máquina remota aceitará a conexão, nesse exemplo está sendo usado o serviço nativo do *Windows*.

Feita essa configuração, o usuário deve estabelecer uma conexão com a máquina na rede DMZ e mantê-la ativa para então realizar uma conexão por *Remote Desktop* para *localhost* na porta 3390, como na figura 6. Assim será efetuada a comunicação remota.


Figura 6 – Acesso remoto por *Remote Desktop*

As figuras 7 e 8 mostram com detalhes como é estabelecida essa comunicação. A figura 7 mostra o estabelecimento da comunicação via SSH com a máquina na rede DMZ.

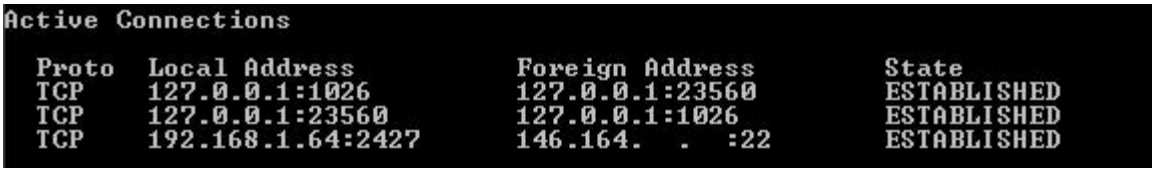



Figura 7 - estabelecimento SSH *prompt Windows*

Já a figura 8 mostra a comunicação via SSH mantida, e o estabelecimento de uma comunicação local com na porta escolhida na configuração.

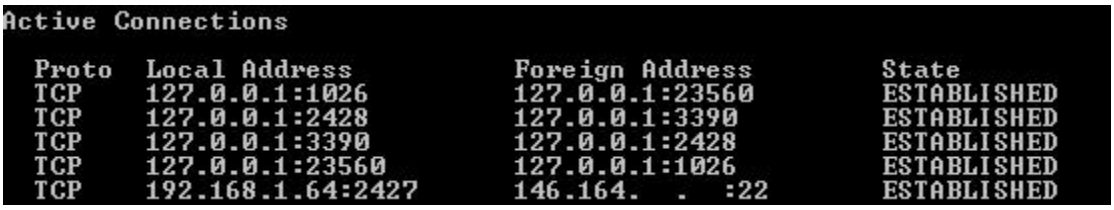



Figura 8 - Conexão SSH e *Remote Desktop* porta 3390 do endereço de *localhost*

Feito isso o acesso será realizado. É possível verificar ambas as conexões estabelecidas na máquina na DMZ. Uma conexão externa na porta 22, e uma conexão interna da máquina na DMZ com a máquina na rede interna com IP invalido na porta 3389. A verificação dessas conexões pode ser com o comando *netstat –n.*

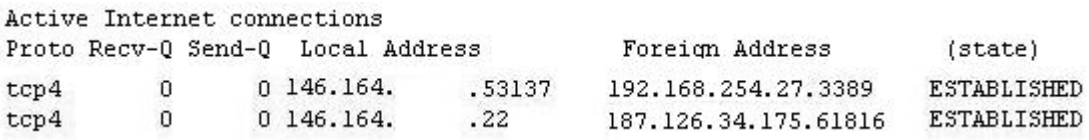



Figura 9- estabelecimento SSH e servidor *Remote Desktop*.

A grande vantagem do uso de acesso remoto por tunelamento é a não exposição das máquinas que possuem essa disponibilidade. Pois em um *scanning,* como no exemplo com o programa *Nmap,* não serão encontradas informações sobre a possibilidade de comunicação externa com essas máquinas, porque essa disponibilidade de conexão só é possível a partir da máquina na rede DMZ.

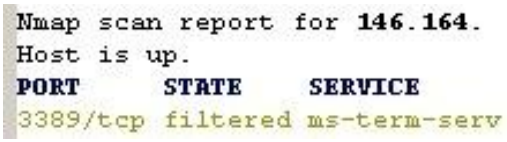



Figura 10 – *Scanning* não encontra a porta aberta.

### 3.2.3 **Segurança em Redes sem Fio**

Outro ponto cuja necessidade de segurança é alta são as redes *Wireless.* A negligência da segurança desse tipo de comunicação pode causar uma grande falha de segurança, pois como esse tipo de comunicação muitas vezes extravasam os limites

das construções, invasores podem se beneficiar dessa “vulnerabilidade” para obter acesso à rede.

Visando ter uma maior segurança também nesse quesito, cada VLAN terá seu próprio *access point*, essa escolha foi feita devido a alguns usuários trabalharem com computadores portáteis pessoais. Além disso, o acesso a ela será controlado por um servidor *Radius*. Esse sistema é utilizado para prover uma autenticação centralizada em redes, assim qualquer usuário que queira acessar a rede *Wireless* precisará de uma conta e senha.

Além da necessidade de autenticação para o acesso, a segurança da comunicação será feita pelo padrão 802.11i, conhecido popularmente como *Wi-Fi Protected Access 2* (WPA2). Esse padrão tem a função de tornar a comunicação relativamente imune à decodificação. Essa segurança é possível pela criptografia inserida na comunicação pelo WPA2. A criptografia realizada por esse padrão usa uma séria de troca de chaves de criptografia que torna a comunicação muito segura.

Então tomando os cuidados citados, pode-se garantir um nível de segurança aceitável a uma rede privada.

## 3.3 SEGURANÇA DOS SERVIÇOS E SISTEMAS

A segurança dos serviços e sistemas é uma área extremamente crítica, onde se requer muito estudo e trabalho, pois a configuração dos servidores e seus serviços devem ser realizados sempre visando obter o máximo de segurança.

Na estrutura dos laboratórios são necessários alguns serviços como o de aplicações *Web*. Estas aplicações são divididas em *Web sites* gerenciais, que contêm informações internas, e só devem ser acessadas por funcionários e *Web sites*

institucionais, que são as páginas de divulgação dos trabalhos executados pelos laboratórios. Esses servidores usarão a aplicação *Apache*, que serve de suporte para aplicações *Web*, rodando sobre os protocolos HTTP e HTTPS. Segundo o *site netcraft* [10], o *Apache* [17, 18, 19] é o sistema mais usado para esse tipo de função. Neste mesmo servidor será preciso ter também suporte a PHP [12], que é uma linguagem de programação muito usada na construção de aplicações *Web*.

Algumas das aplicações que serão usadas, como os gerenciadores de conteúdo, têm a necessidade da criação de bancos dados. Então para atender essa necessidade, será usado o sistema gerenciador de banco de dados MySQL [14]. Por questões de segurança será usado um servidor exclusivo para essa tarefa e esse servidor estará em uma VLAN exclusiva. A opção de colocar os bancos de dados em um servidor separado se dá porque as duas aplicações referenciadas anteriormente apresentaram uma série de problemas de segurança, tornando-as vulneráveis no caso de não haver um controle rigoroso quanto à descoberta e solução dessas vulnerabilidades. Já a opção de mantê-lo em uma VLAN exclusiva, tem por objetivo minimizar ataques direcionados ao servidor, seja internamente ou externamente. Assim mantendo as aplicações sempre atualizadas e construídas com boas práticas é possível minimizar a possibilidade de ocorrer um ataque bem sucedido.

Outro serviço importantíssimo para o bom funcionamento da rede é o *Dinamic Name System* (DNS) [21], serviço responsável pela tradução de nomes de domínios em endereços IP. Esse serviço será dividido em duas *views*, interna e externa. Essa solução será adotada, para que máquinas e servidores que não devem ser conhecidos externamente tenham suas informações propagados para fora da rede.

A última aplicação é a mais crítica para estrutura dos laboratórios, o servidor de e-mail. Esse servidor rodará duas aplicações, uma é o agente de correio muito usado e conhecido pela sua robustez e segurança que é o *Postfix* [32]. O *Postfix* será o responsável pela troca de mensagens entre servidores. Porém para se ter acesso às mensagens precisa-se de outro serviço que faça a comunicação entre o *Postfix* e a aplicação que permitirá o acesso das mensagens pelos usuários rodando neste mesmo servidor, que é o *Dovecot* [32]. Essa aplicação será responsável por intermediar a autenticação dos usuários, pela formatação das caixas de mensagens, e por definir qual o protocolo de acesso às mensagens será usado. Os protocolos que podem ser usados são: IMAP (*Internet Message Access Protocol*) e POP3 (*Post Office Protocol*). O protocolo que será usado é o IMAP, essa escolha foi feita por levar em contra o seu método de funcionamento: somente as mensagens que serão lidas são baixadas para acesso do usuário. Já o POP3 faz *download* de todas as mensagens a cada conexão, mesmo as mensagens que não são necessárias, gerando um tráfego muitas vezes desnecessário, o que pode gerar o risco de todas as mensagens serem baixadas e apagadas do servidor, no caso de uma configuração mal feita em um gerenciador de e-mails.

Uma vez que foi visto um pouco sobre cada aplicação que será usada, agora serão abordadas algumas configurações importantes para um funcionamento seguro dessas aplicações.

### 3.3.1 **Segurança em Servidores *Web***

Começando pelo servidor *Web*, como foi dito anterior é muito importante ter atenção nos servidores que fornecem esse tipo de serviço que são uns verdadeiros

parques de diversão para hackers se não estiverem bem configurados, atualizados e protegidos. Pode-se constatar isso visto o número de vulnerabilidades encontradas nas aplicações usadas nesses servidores e com dados do *site zone-h.org* [11]. Neste site pode-se verificar a quantidade de servidores que são invadidos por minuto. Só no momento do acesso para essa consulta foram invadidos mais de 60 *sites*. Assim pode-se ter a dimensão do quanto é importante cuidar da segurança de servidores *Web*.

3.3.1.1 Segurança no *Apache*

Essa aplicação fornece o serviço base desse servidor, então torna-se necessário cuidar de sua segurança. O principio da segurança está em não dar informações, para isso tem-se um par de configurações básicas que são: *ServerSignature Off* e *ServerTokens Prod,* que aliadas ao uso de uma opção simples do módulo *mod_security* que é a *SecServerSignature,* realizam a função de tornar a aplicação totalmente obscura, ou seja, nenhum tipo de informação sobre a mesma é divulgada, seja acessado um diretório inexistente ou fazendo um *scanner* no servidor. As figuras 11 e 12 mostram o resultado dessas configurações. O lado esquerdo exemplifica o não uso das opções e o lado direito o uso das três funcionalidades citadas.

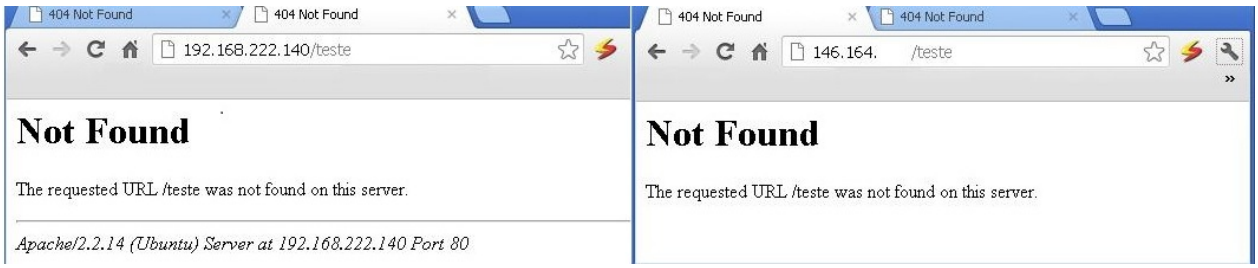



Figura 11 – Diferença em uma tentativa de acesso a um diretório não existente.

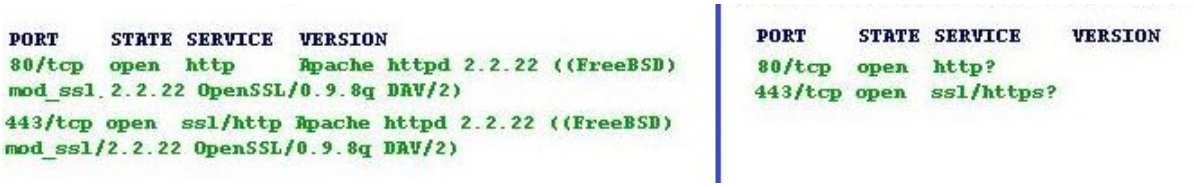



Figura 12 – Diferença em um *scanning*.

Outra configuração simples que traz segurança, principalmente na não exposição dos diretórios do *site*, é só permitir o acesso a diretórios específicos. Esse tipo de configuração é importante para controlar quais os diretórios que serão acessíveis. Um exemplo dessa configuração é vista no quadro 6.

Quadro 6 - Configuração de permissão de diretório no *Apache*

```
<Directory />
  AllowOverride None
  Order deny,allow
  Deny from all
</Directory>
```

```
<Directory "/usr/www/meusite">
  Order allow,deny
  Allow from all
</Directory>
```

Conforme exemplificado, somente no diretório desejado é permitido o acesso.

Muito importante também é sempre analisar os módulos instalados, deixando ativo somente os realmente necessários. Para realizar esse tipo de configuração é necessário conhecer bem o ambiente que se está trabalhando. Como é mais simples desabilitar módulos do que adicionar novos módulos após a instalação, é mais prático realizar a instalação como todos os módulos disponíveis e posteriormente os desabilitá-los. Para desabilitar um módulo basta comentá-lo no arquivo de configuração do *Apache* e reiniciar o serviço.

Um módulo que pode ser desabilitado é o *mod_autoindex*, pois como sua função é gerar automaticamente um *index* com a lista de todos os arquivos e pastas de um diretório. Como não se quer que isso ocorra, esse módulo pode ser desabilitado.

Para finalizar, um módulo que é importante para o fortalecimento de um servidor *Web* é o *mod_security*. Esse módulo tem uma série de funcionalidades muito úteis para proteger servidores *Web*, pois ele funciona como um *firewall* para essa aplicação. Essa proteção é feita por meio de regras escritas por expressões regulares, podendo ser adaptadas às necessidades específicas de um ambiente. Assim um servidor com esse módulo instalado terá sua capacidade de lidar com ataques fortalecida.

3.3.1.2 Segurança do PHP

O PHP é a principal linguagem de programação que será usada no desenvolvimento das aplicações. Além das boas práticas seguidas para construir uma programação segura, existem configuração que são úteis para essa tarefa.

Para tornar o *kernel,* código central do PHP seguro, algumas configurações são recomendadas. Algumas delas são simples como: somente deixar módulos necessários ativos, restringir informações sobre versão utilizada, e não exibir os erros de programação mas os registrar em arquivos de *log*.

Existem também opções mais relevantes, muitas delas se tornaram padrão e estão disponíveis com explicações no próprio arquivo de configuração, mas configurações como a permissão de uploads de arquivos *file_uploads* e a execução de dados a partir de outros *sites allow_url_fopen*, por padrão são habilitadas. Caso não haja necessidade do uso dessas funções o ideal é desabilitá-las ou só habilitá-las quando necessário.

Outras configurações importantes que visam minimizar a ocorrência de um ataque de DOS são: *max_execution_time,* que determina o tempo máximo que um script será executado; *max_input_time, que* determina o tempo máximo de análise de dados recebidos; e *memory_limit,* que define a quantidade de memória que poderá ser usada por um *script*. Deve-se analisar o uso dessas funções de acordo com as necessidades do ambiente, mas com os devidos cuidados, pois uma má configuração das mesmas pode gerar brechas para um ataque de DOS.

Para finalizar as configurações de segurança do PHP será abordado o uso de um mecanismo que tem a função de proteger as aplicações escritas em PHP e o seu *kernel.* Esse mecanismo é o *Suhosin* [15]. Esse módulo é dividido em duas partes, cabendo ao administrador definir se usará ou não ambas, pois as mesmas são independentes. A primeira parte funciona como um *patch* para proteção do *kernel* do PHP e pode ser adicionado no momento da instalação do mesmo. A segunda parte é uma extensão com várias proteções para aplicações PHP e esse é instalado separadamente, posteriormente a instalação do PHP.

Com essas dicas pode-se melhorar sensivelmente a segurança de um servidor *Web*. Esse tipo de servidor deve ter um acompanhamento e ser atualizado continuamente, pois novas vulnerabilidades são encontradas com frequência.

### 3.3.2 **Segurança do Servidor de DNS**

Nessa sessão serão abordadas algumas configurações de segurança de um serviço muito importante que é o DNS, serviço fundamental para o funcionamento do acesso à internet e do servidor de *e-mail*. Seu funcionamento consiste em arquivos com informações de nomes e os respectivos endereços IP das máquinas de uma rede.

Esses arquivos são denominados zonas, todas as informações sobre um domínio estão contidas em uma zona.

As zonas se dividem basicamente em zona direta, que é responsável pela conversão de nomes em endereços IP, e zona reversa que é responsável pela tradução de um endereço IP em um nome. Para o bom funcionamento da comunicação, por exemplo, de servidores de *e-mail*, além da zona direta é fundamental a configuração da zona reversa.

A segurança desse serviço é fundamental, seguindo a diretriz de não dar informações sobre o serviço, usa-se a configuração *Unknown* para a função *version.* Assim seja por uma consulta com o uso de um *software* ou por um *scanning* a versão utilizada dessa aplicação não será informada.

Uma funcionalidade que traz muita segurança, sobre tudo quanto a não divulgação de informações sobre uma rede, é a possibilidade de se configurar o DNS para trabalhar com *views*. Elas determinam quais máquinas têm permissão para realizar consultas sobre uma zona.

Por questão de segurança a configuração será dividida em duas *views*. A primeira é a *view* interna, onde ficarão contidas todas as informações que devem ser conhecidas somente pela rede interna. Nela também serão definidos quais os clientes que terão permissão para realizar consultas recursivas, que são as consultas feitas a fim de acessar um servidor externo. Essa definição é aplicada pelas seguintes configurações: *allow-query {* clientes*; }*, que determina quais os clientes podem fazer consultas nessa *view*. O parâmetro clientes faz referência a ACL clientes (quadro 7), que determina quais são as máquinas que podem fazer as consultas. A outra opção é *recursion yes*, que habilita a capacidade de realizar consulta recursivas.

Quadro 7 – Configuração de ACL no DNS

```
acl clientes {
  localhost;
  146.164.xxx.0/24;
  192.168.254.0/24;
};
```

Como esse servidor também fará a função de um DNS autoritativo, usado para divulgar informações necessárias para acesso a partir da rede externa, a segunda *view,* denomina externa será usada para realizar essa função. Nessa *view* é preciso permitir que qualquer máquina de qualquer rede realize requisições, mas que não façam consultas recursivas e não será enviada nenhuma resposta a essas consultas. Só serão respondidas as consultas sobre as zonas cujo servidor tem autoridade.

Como também não há necessidade de realizar transferência de zonas e realizar *updates* automáticos, essas opções não serão permitidas em nenhuma das *views*. O quadro 8 mostra um exemplo de configuração das opções citadas.

Quadro 8 – Exemplo de configuração de *view* no DNS

```
view "externa" {
  match-clients { any; };
  recursion no;
  additional-from-auth no;
  additional-from-cache no;
  allow-transfer { none; };
  allow-update { none; };
};
```

Para finalizar existe um mecanismo de segurança que pode ser implementado em qualquer servidor de DNS, que é o DNSSEC [25, 26, 27]. A função dessa tecnologia é tornar a troca de informação entre os servidores de DNS mais segura, mas para isso é preciso uma ampla implementação da mesma. O DNSSEC funciona da seguinte

maneira: um servidor terá um par de chaves de criptografia, uma chave privada usada para assinar as informações geradas, e uma chave pública que será compartilhada e deve ser usada pelos outros servidores para validar essas informações. Assim todas as informações geradas por um servidor que tenha esse mecanismo terão suas informações autenticadas, garantido que as informações são válidas.

Esse mecanismo é muito útil contra uma das principais ameaças da internet que é o *spoofing* de endereços. Como todas as informações geradas por um servidor são assinadas, uma rede que utilize um servidor recursivo com esse mecanismo estará imune a esse problema.

### 3.3.3 **Segurança no Servidor de E-mail**

Na estrutura dos laboratórios o serviço de maior importância para os usuários é o *e-mail*. Então além de serem abordadas configurações de segurança, serão também verificados mecanismos  para o de controle de *spans*, que são as mensagens recebidas sem requisição.

Iniciando com técnicas de segurança para o acesso, como esse serviço vai ser usado principalmente por *softwares* gerenciadores e *smartphones,* é necessário que os protocolos que serão usados para essa comunicação tenham criptografia. Cada um desses protocolos será controlado por uma das aplicações que são o *Postfix* e o *Dovecot*. Dessa forma garante-se uma comunicação segura entre o usuário e o servidor.

O *Postfix* será configurado com dois mecanismos adicionais para ter-se o nível de segurança necessário. O primeiro é o SASL [28] *(Simple Authentication and Security Layer*), que é o protocolo que irá adicionar autenticação à comunicação do cliente com o servidor permitindo que esses e-mails sejam acessados de gerenciadores. O outro

mecanismo é o TLS [29] (*Transport Layer Security*) que irá adicionar encriptação ao sistema de autenticação. Gerando assim um sistema de *e-mail* seguro para acesso remoto. O quadro 9 exemplifica essas configurações.

Quadro 9 – Configurações do *Postfix* para acesso externo seguro

| | |
|---|---|
| Smtpd_sasl_type = dovecot | Determina o serviço que autenticará os clientes |
| Smtpd_sasl_auth_enable = yes | Habilitá a autenticação dos Usuários |
| Broken_sasl_auth_clients = yes | Opção usada para comunicação com clientes de e-mail |
| Smtpd_sasl_authenticated_header = yes | Habilita a assinatura do cabeçalho das mensagens |
| Smtpd_sasl_path = private/auth | Caminho do socket de comunicação para autenticação, troca de informações das autenticações |
| smtp_use_tls = yes | Habilitá a criptografia usando TLS |
| Smtpd_use_tls = yes | Habilitá a criptografia usando TLS |
| Smtpd_tls_auth_only = yes | Fornecerá criptogrfia somente à clientes autenticados |
| Smtpd_tls_received_header = yes | Requisição de informações do cliente |
| Smtpd_tls_session_cache_timeout = 3600s | Tempo de duração de uma sessão |
| Tls_random_source = dev:/dev/urandom | Geração de códigos de criptografia |
| Smtpd_tls_loglevel = 1 | Nível de informção do Log |
| Smtpd_tls_key_file = /../mail/mail.key | Caminho da chave privada |
| Smtpd_tls_cert_file = /../mail/mail.crt | Caminho da chave pública |

Com essas configurações todo o processo de gerenciamento da autenticação dos usuários e de criptografia do transporte das mensagens será feito pelo *Postfix*.

No quadro 10 estão as configurações necessárias para que o *Dovecot* formate as caixas de mensagens com o protocolo desejado e a configuração do uso dos mecanismos adicionados aos *Postfix* para realizar a comunicação com os usuários.

Quadro 10 – Configuração do *Dovecot* para acesso externo seguro

| | |
|---|---|
| protocols = imap imaps | Define o tipo de protocolo será usado para as mensagens |
| ssl = Yes | Habilita o suporte a ssl |
| socket listen | Habilita o uso do método de autenticação usado pelo Postfix |
| path = /var/spool/postfix/private/auth | Caminho do socket de autenticação |
| mode = 0660 | permissão desse socket |
| user = postfix | Usuário dono desse arquivo que funciona como socket |
| group = postfix | Grupo dono desse arquivo que funciona como socket |

Além das configurações de segurança para acesso dos usuários, deve-se fazer configurações de segurança a nível de troca das mensagens, essas configurações usam como parâmetros alguns dos campos das mensagens. Tais configurações visam sobretudo o bloqueio de *spans*, que podem ser desde uma simples propaganda até códigos maliciosos que podem infectar toda uma rede.

Essas configurações e combinações de configurações são feitas com o uso das macros disponibilizadas pelo *Postfix*. Algumas macros importante são: *smtpd_client_restrictions*, que serão as regras aplicadas no momento do estabelecimento da conexão; *smtpd_helo_restrictions*, regras aplicadas ao comando *HELO*, a primeira fase após o estabelecimento da conexão; *smtpd_sender_restrictions*, que são as regras aplicadas em relação ao campo *From;* e *smtpd_recipient_restrictions***,** que são regras aplicadas em relação ao campo *To.* Usando essas quatro macros pode-se realizar uma série de controles.

Na macro *smtpd_client_restrictions* é interessante ter as opções, descritas no quadro 11.

Quadro 11 – Configuração do *Postfix* para bloqueio de *spans* 1

| | |
|---|---|
| permit_mynetworks, | Permitir ips definidos como ips das minhas redes |
| permit_sasl_authenticated, | Permitir clientes autenticados |
| reject_unknown_client, | Rejeitar quando ocorrer inconsistência no mapeamento do nome da máquina pelo DNS |
| reject_unauth_pipelining, | Rejeita o envio de um mensagem de uma única vez, quer dizer não aguarda as respostas do servidor. |
| reject_multi_recipient_bounce, | Rejeita o envio de um mensagem de com emissor nulo e múltiplos destinatários |
| reject_rbl_client | Faz uso de *blacklist* para bloquear emissores |
| header_checks | Regras de bloqueio feitas por expressões regulares. Com essas regras pode-se bloquear qualquer coisa nos campos From, To e Subject, além de poder bloquear facilmente arquivos por extensões |
| permit | Permite a conexão, esse opção sempre deve ser coloca ao fim de uma lista de restrições |

Para a ativação de algumas opções da macro *smtpd_helo_restrictions* é necessário a configuração do parâmetro *smtpd_helo_required*. Algumas opções dessa macro estão no quadro 12:

Quadro 12 – Configuração do *Postfix* para bloqueio de *spans* 2

| | |
|---|---|
| reject_invalid_helo_hostname | Rejeita quando for informado um nome invalido |
| reject_non_fqdn_helo_hostname | Rejeita caso não haja um rementente sem um domínio válido |
| reject_unknown_helo_hostname | Rejeita quando um servidor não tem configurado adequandamente um nome ou MX (mail exchanger) |

Na macro *smtpd_sender_restrictions*, algumas opções úteis são (quadro 13) :

Quadro 13 – Configuração do *Postfix* para bloqueio de *spans* 3

| | |
|---|---|
| reject_non_fqdn_sender | Rejeita caso o emissor não possua um domínio válido |
| reject_unknown_sender_domain | Opção util para servidores de relay, rejeita caso o destino não tenha um nome válido ou um MX |
| warn_if_reject, | Caso uma mensagem seja rejeita fará um alerta |
| reject_unverified_sender | Rejeita caso o destino seja um endereço de bounce ou um endereço inválido |

A última macro é a *smtpd_recipient_restrictions*, e as regras úteis são (quadro 14):

Quadro 14 – Configuração do *Postfix* para bloqueio de *spans* 4

| | |
|---|---|
| reject_unauth_destination | Usada para fechar o relay do servidor, só permitindo os previamente autorizados. |
| reject_non_fqdn_recipient | Rejeita caso o receptor não possua um domínio válido |
| reject_unknown_recipient_domain | Opção util para servidores de relay, rejeita uma requisição quando o servidor não é destino final para o receptor e o campo To não tem um nome válido ou um MX |
| reject_unverified_recipient | Rejeita caso o campo To seja um endereço de bounce ou um endereço inválido. |

Além das configurações mostradas anteriormente, tem-se mais algumas que ajudam no controle *Anti-Spam* e não estão relacionadas a nenhuma das macros citadas. Essas configurações consistem em forçar o envelopamento correto dos *e-mails* com a opção *strict_rfc821_envelopes* e também ao uso de *softwares* antivírus e *Anti-Spam*.

Com esse conjunto de configurações torna-se mais seguro o servidor de *e-mail*. Mas é muito importante também orientar os usuários que usem o *e-mail* de forma consciente, pois eles também fazem parte da corrente de segurança, orientações simples como não clicar em qualquer *link* recebido por um *e-mail* e não abrir qualquer arquivo recebido, entre outros cuidados, pois os filtros *Anti-Spam* têm a função de minimizar a ocorrência dessas pragas, mas dificilmente conterão todas.

### 3.3.4 **Segurança dos Servidores**

A segurança e estabilidade dos sistemas operacionais que rodaram os serviços e aplicações citadas anteriormente são muito importantes, por isso a escolha desse

sistema deve ser feita com cuidado. Por questões de segurança, estabilidade e usabilidade foi escolhido o sistema *FreeBSD* para ser a plataforma desses servidores. O *FreeBSD* é um sistema operacional de código livre baseado na arquitetura *Unix*, que foi desenvolvido e é mantido pela universidade de *Barkely*.

O que é preciso para ter um sistema seguro é mantê-lo atualizado e aplicar algumas diretrizes de segurança, como as que serão discutidas a seguir. A maioria dos sistemas operacionais, em sua configuração padrão, têm uma série de aplicativos sendo executados sem necessidade. Então para evitar que aplicativos não utilizados gerem falhas de segurança, será realizada busca e desativação desses serviços. Essa é uma atitude que deve ser tomada, pois esses serviços podem conter vulnerabilidades que podem colocar o sistema em risco.

O *FreeBSD* em sua instalação padrão tem os seguintes serviços sendo executados: *sendmail*, que é um servidor de e-mail; e *syslog*, que é o aplicativo que gerencia os *logs* dos serviços instalados no sistema. Como o *sendmail* não é o aplicativo escolhido para essa função e os servidores não precisam desse serviço, pode-se desabilitá-lo permanentemente através dos parâmetros *sendmail_enable="NO",* *sendmail_submit_enable="NO*" e *sendmail_outbound_enable="NO".*

Já o serviço *syslog* não pode ser desabilitado, mas pode funcionar de modo mais seguro com a seguinte configuração *syslogd_flags="-ss".* Assim o *syslog* será executado somente com acesso local, pois o seu padrão é ser executado com possibilidade de acesso remoto. Caso o sistema em questão não seja um servidor de *log* a funcionalidade do acesso remoto não é necessária.

Outras configurações, essas a nível dos protocolos da arquitetura TCP-IP são: *icmp_drop_redirect="YES",* para rejeitar qualquer pacote que tente indicar uma nova rota a ser seguida, opção útil principalmente para roteadores; *net.inet.tcp.always_keepalive=1*, usada para verificar a existência de conexões encerradas que não foram finalizadas, para então as encerrar; e *net.inet.tcp.drop_synfin=1*, usada para rejeitar qualquer pacote que tenha as *flags* TCP Syn e Fin ativas simultaneamente, pois essa técnica é muito usada em ataques.

Opções como *net.inet.tcp.blackhole=2* e *net.inet.udp.blackhole=1*, que são usadas para bloquear qualquer tentativa de conexões às portas fechadas, que aliadas às opções *net.inet.tcp.log_in_vain=1* e *net.inet.udp.log_in_vain=1*, que são usadas para logar as tentativas de conexão a portas fechadas, são muito úteis para identificar *scanning* aos servidores.

Uma opção útil para minimizar os impactos de um ataque DOS é a *net.inet.tcp.msl*. Essa opção tem a função de determinar o tempo de espera máximo, em milissegundos, de um ACK em resposta de um SYN-ACK ou FIN-ACK, o padrão do sistema é de 30000 ms, um tempo bastante alto para a as latências das redes atuais, então pode-se avaliadar a redução desse tempo. Outra opção que também fortalece o servidor contra os ataques de DOS é *kern.ipc.somaxconn*, ela limita o número máximo de *sockets* disponíveis para conexões, o padrão do sistema são 128. Isso poderia ser um facilitador para um ataque DOS com *syn-flood* caso não fosse implementado por padrão a técnica de *syncookie* [36]. A função dessa técnica é, através de um cálculo secreto, especificar um número de sequência no ato do recebimento de um pacote com as *flags* SYN-ACK, usado para iniciar uma conexão, e o enviar para o emissor. Dessa forma só caso seja recebido o ACK de resposta válido será alocado o recurso para a

conexão. Então o parâmetro da opção *kern.ipc.somaxconn* pode ser aumenta somente em caso de esgotamento de *sockets* por uso válido, pois dificilmente serão esgotados por ataques.

Além das configurações citadas existem outras configurações que visam tornar o sistema mais robusto. Algumas opções são implementadas por padrão tornando o *FreeBSD* um sistema bastante seguro, algumas delas são: *net.inet.icmp.bmcastecho*, que tem a função de não responder a ICMP originados de endereço de broadcast, isso evita o uso de máquinas com esse sistema participem de ataques do tipo *icmpsmurf*, que são ataques que usam máquinas que respondem a esse tipo mensagem; *net.inet.icmp.maskrepl=0*, que têm a função de não dar nenhuma informação sobre a máscara de rede usada; e as opções *net.inet.ip.sourceroute=0* e *net.inet.ip.accept_sourceroute=0*, têm a função de evitar a ocorrência de um ataque de *spoofing*.

Com isso foram abordados parâmetros e configurações de segurança para a rede como um todo e os serviços disponibilizados.

# 4 **MONITORAMENTO**

Uma vez que foram verificadas algumas formas de se ter segurança na rede, nos sistemas e serviços, tratar-se-á do monitoramento dessa infraestrutura. O monitoramento será feito com o intuito de verificar possíveis vulnerabilidades, ataques oriundos da rede externa ou interna, atividade dos serviços e estado da rede de forma geral.

## 4.1 MONITORAMENTO DE SEGURANÇA

Esse monitoramento será feito em todos os servidores e *switches,* e em máquinas de determinados setores. O monitoramento da segurança será feito por *software* de análises como o IDS, mensagens *Traps* usando o protocolo SNMP, por *softwares* que realizam checagens de *logs,* e verificação de vulnerabilidade em aplicações instaladas nos servidores.

### 4.1.1 **IDS**

Uma boa ferramenta para verificar tentativas indevidas de acesso é o *Intrusion Detection System* (IDS), que é um sistema cuja função é observar todos os pacotes trafegando pela rede, para então aplicar regras de assinatura de ataques ou por análise do comportamento, determinar se os pacotes são ou não suspeitos. Os IDSs realizam um método de monitoração passivo, ou seja, só geram alertas, nenhum tipo de ação será aplicada pelo IDS, sendo assim caberá ao administrador avaliar o alerta e tomar as devidas medidas.

Como um IDS trabalhando no modo de *network* IDS (NIDS), que irá monitorar toda a rede, será gerada uma grande quantidade de informação, por isso o modo de funcionando *host* IDS (HIDS) se mostra mais interessante. Nesse método de funcionamento será instalado um agente nos *hosts* e então serão monitorados somente

os pacotes referentes às determinadas máquinas. Uma das vantagens desse método é a versatilidade de operar ao nível do sistema operacional, tendo uma gama maior de informações sobre os possíveis ataques [6, 8].

O uso dessa ferramenta com a configuração sugerida tem a vantagem de trabalhar com um volume de informação e alertas menor, podendo realizar uma análise e obter uma solução mais eficaz.

### 4.1.2 **Alertas via SNMP**

Outro mecanismo de alerta que pode ser aplicado no monitoramento da segurança da rede interna, focado nos equipamentos da rede como os *switches*, é o envio de alertas SNMP. Por meio de alertas através de *Traps*, que são as mensagens enviadas do agente para o servidor no caso de ocorrer algum tipo de falha. Essas mensagens podem ser geradas, por exemplo, na ocorrência de alguma violação de segurança do mecanismo de *Port Security* em um *switch*. A figura 13 mostra o recebimento dessa informação pelo *software* MIB Browser.

Com essa informação o administrador pode acessar o equipamento e verificar o que gerou esse alerta.

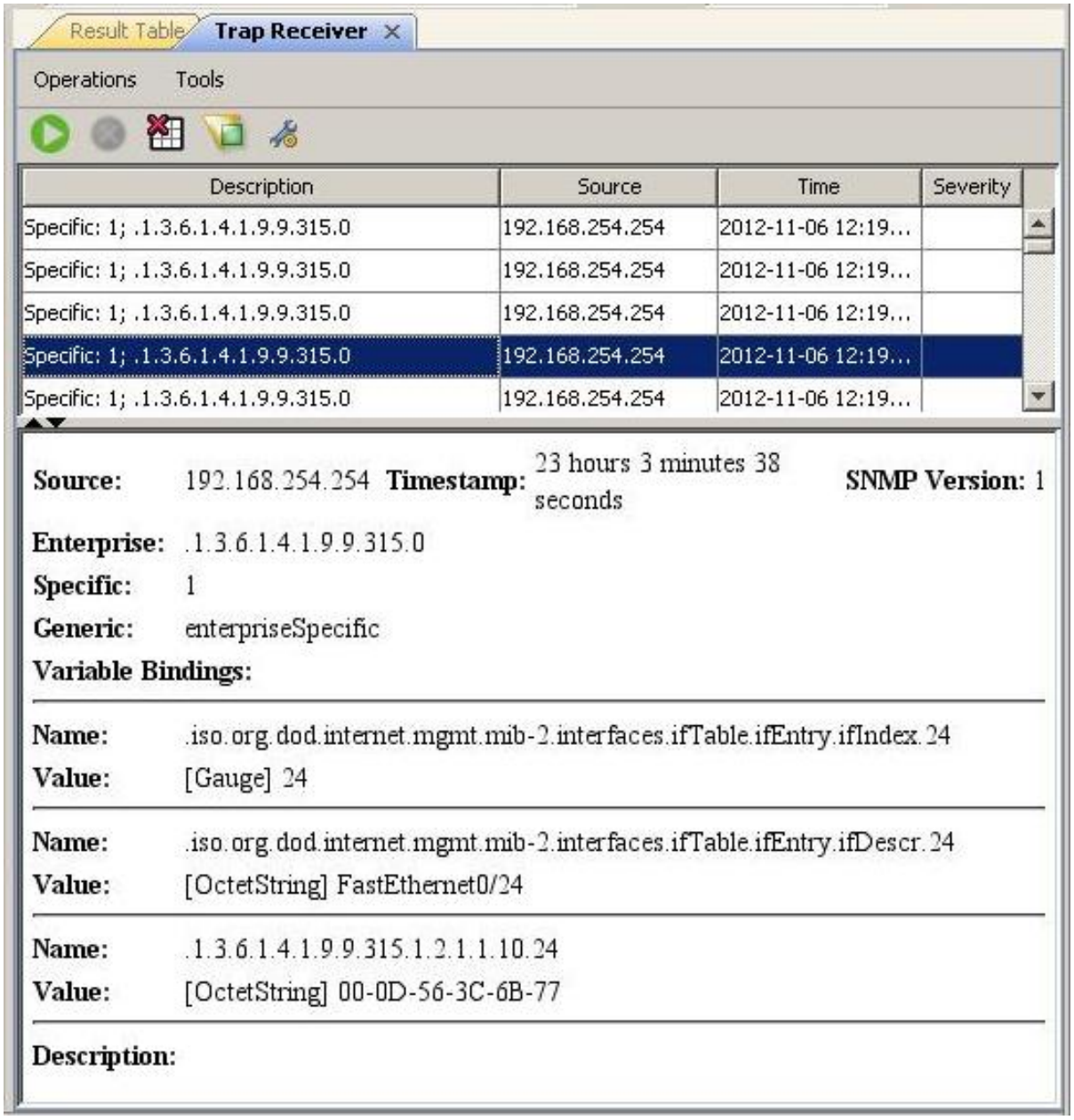



Figura 13 – *Software* MIB *Browser*

### 4.1.3 **Monitoramento de Logs e Verificação de Vulnerabilidades**

Para essa tarefa foram usados dois *softwares*: um para checagem de logs que é o *Logcheck* [39], e outro para verificar vulnerabilidades nos *softwares* instalados que é o *Portaudit* [41], ambos geram alertas por *e-mail*. O *Logcheck* como o nome já sugere, tem seu funcionamento baseado na checagem dos arquivos de *log*. Essas checagens

são feitas baseando-se em *scripts* escritos por expressões regulares que buscarão determinadas ocorrências nos arquivos de *log*, então caso seja encontrada alguma ocorrência será gerado um alerta.

Na figura 14 pode-se verificar dois desses alertas. O primeiro exemplo é um alerta de uma tentativa de envio de *spam* sendo bloqueado. O segundo exemplo mostra o bloqueio da tentativa de acesso via SSH de um usuário não permitido.

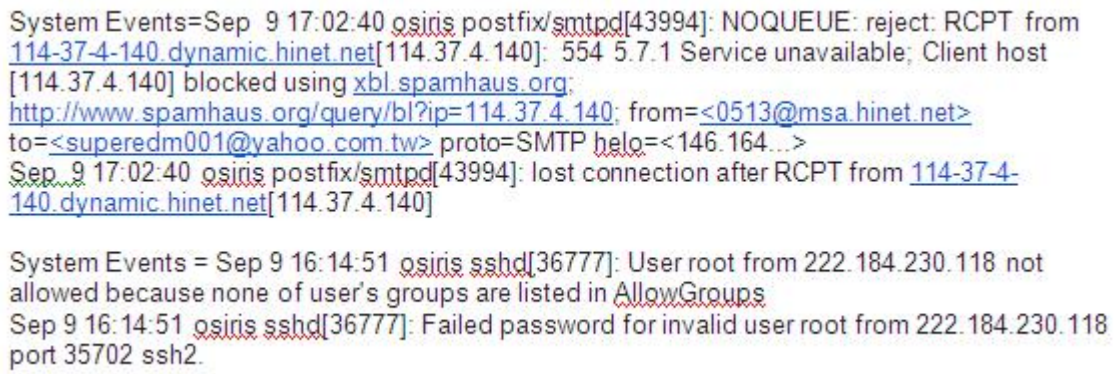



Figura 14 – Informação de um Bloqueio de *spam* gerado pelo *logcheck*

Já o *Portaudit, que é* usado para verificar e informar possíveis falhas de segurança nos *softwares* instalados funciona da seguinte forma: com base nas informações geradas e disponibilizadas pelo suporte do sistema *FreeBSD,* sobre as versões de *softwares* que contenham vulnerabilidades, é realizada uma checagem se algum dos *softwares* instalados no servidor corresponde a alguma das versões relatadas, e então é gerado um alerta sobre esse risco.

Essa verificação pode ser realizada manualmente através do comando *portaudit –Fda*, mas ela também ocorre de forma automática. A informação automática é realizada diariamente e a informação de possíveis vulnerabilidades é gerada através de um alerta por *e-mail.* Na figura 15 pode-se verificar um exemplo desse alerta.


Figura 15 – Informação de vulnerabilidade gerada pelo *Portaudit*

## 4.2 MONITORAMENTO DA REDE

O monitoramento da atividade dos serviços, sistemas, equipamentos e da disponibilidade da rede serão feitos com uso de dois protocolos SNMP, ICMP, através de uma tecnologia desenvolvida pela *Microsoft* para monitoramento de sistemas *Windows* chamada *Windows Management Instrumentation* (WMI), e por checagens usando os protocolos da arquitetura TCP-IP.

### 4.2.1 **Monitoramento dos Serviços**

O monitoramento dos serviços disponíveis será feito com o uso do *Nagios* [41], que é um *software* para gerenciamento que tem a possibilidade de trabalhar com os protocolos SNMP, ICMP e com checagens das portas usadas pelos protocolos da arquitetura TCP-IP. O *Nagios* foi escolhido para esta função devido à simplicidade e praticidade com que se pode fazer customizações para realizar checagens da atividade dos serviços.

Com o *Nagios* é possível fazer uma checagem diretamente na porta de comunicação usada pelo protocolo, sendo assim possível verificar se o serviço está ativo, como demonstrado na figura 16.

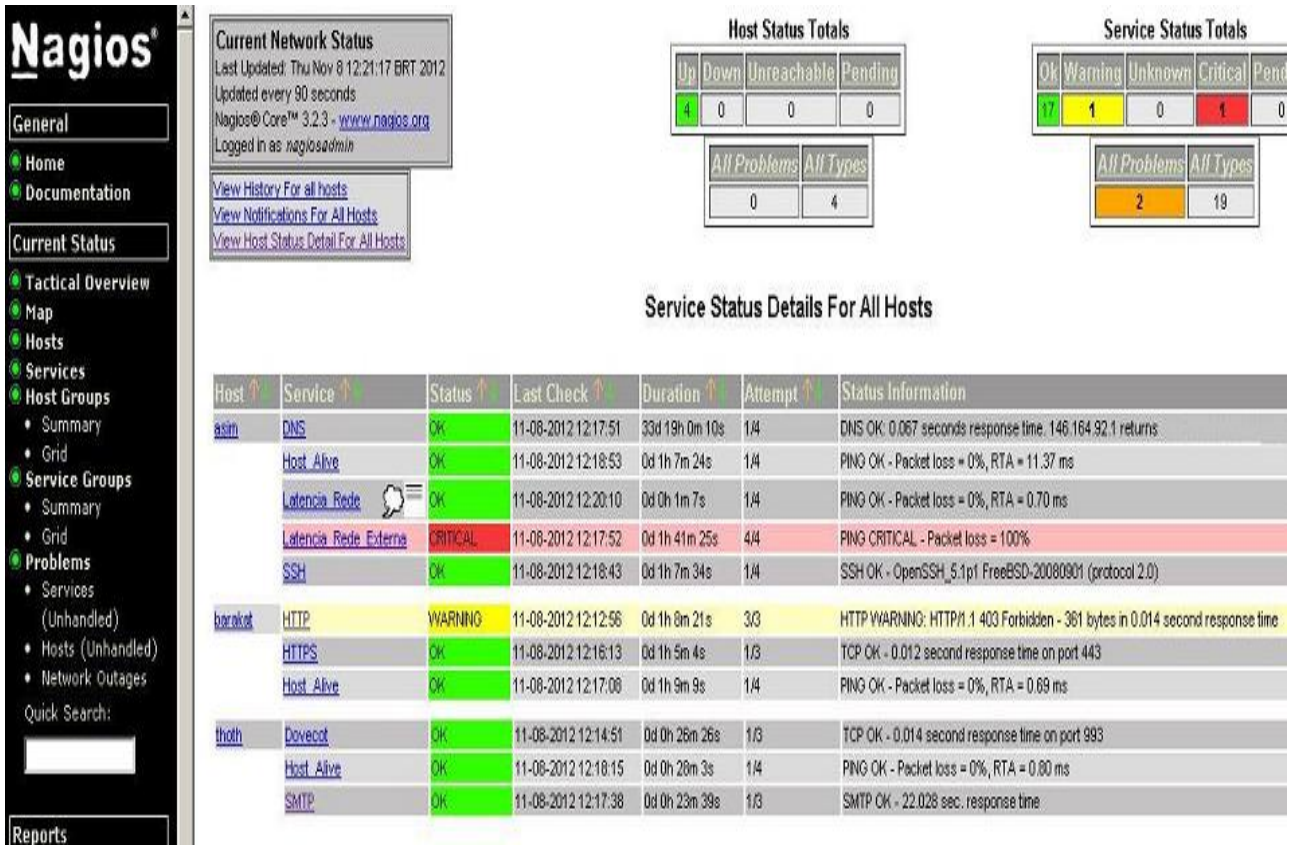



Figura 16 – Gerenciamento do *status* do serviço pelo Nagios

Pode-se verificar na figura 16 que a checagem do estado dos serviços *Dovecot* e HTTPS são realizadas através da verificação da atividade usando o protocolo TCP. A configuração dessas checagens são vistas no quadro 15.

Quadro 15 – Configuração para checagens de portas através do *Nagios*

| Comando | Parâmetros |
|---|---|
| Command_line $USER1$/check_tcp | – H $HOSTADDRESS$ -p 993 –t 60 |
| Command_line $USER1$/check_tcp | – H $HOSTADDRESS$ -p 443 –t 60 |

### 4.2.2 **Monitoramento de Sistemas e Ativos**

Nessa fase do monitoramento será feito o uso de dois *softwares Cacti* e PRTG. O *Cacti* [43] que assim como o *Nagios* é uma excelente ferramenta para monitoramento e será usado para fazer monitoramento dos servidores através de informações via SNMP.

Nos servidores serão monitorados a utilização de memória, utilização do espaço em disco, tráfego de rede e consumo de processamento. Já nos ativos como os *switches* serão monitorados o consumo de processamento e o tráfego rede. As figuras 17 e 18 exemplificam esses monitoramentos.

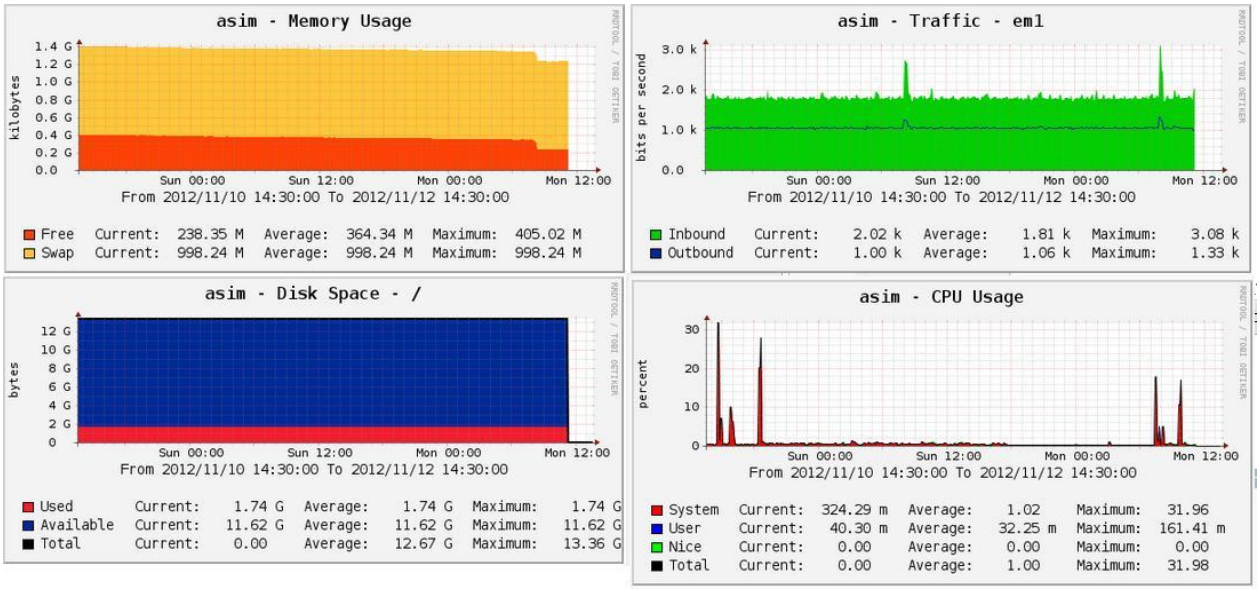



Figura 17 – Monitoramento de um servidor com o Cacti

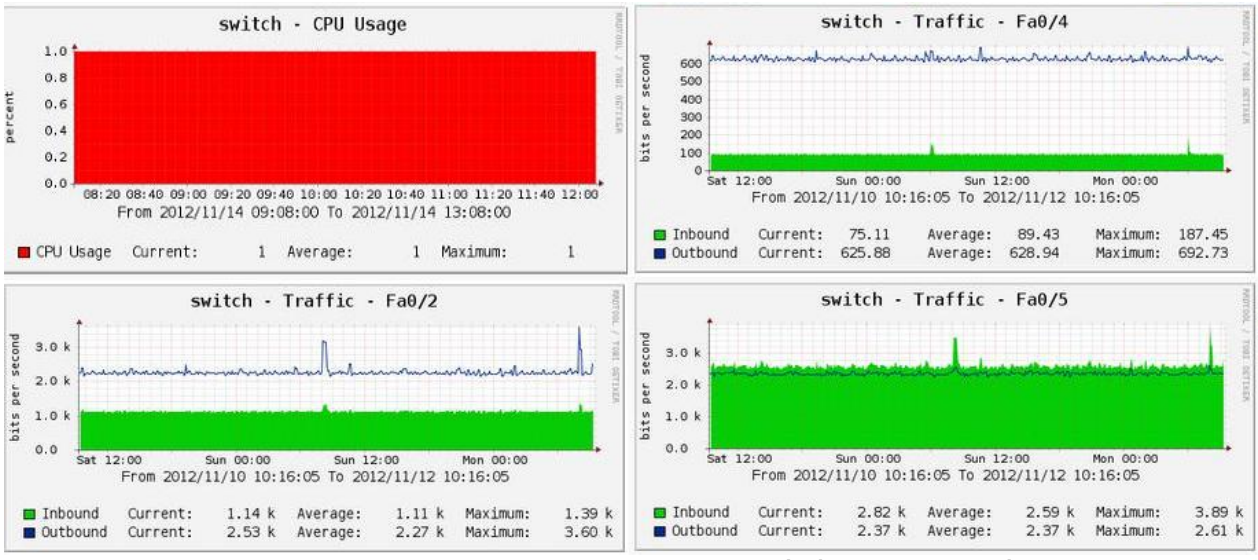



Figura 18 – Monitoramento de um *switch* com o *Cacti*

Para o monitoramento das máquinas cujo sistema operacional é *Windows* será usado o *software* PRTG [44], que além de ser um *software* extremamente simples de

ser utilizado, tem uma grande gama de opções para monitoramento de sistemas *Windows* com o uso da tecnologia WMI. Para realizar o monitoramento com o uso dessa tecnologia basta ter uma conta de administrador da máquina que se deseja monitorar. O mais interessante do uso desse *software* para o monitoramento de sistemas *Windows* é a possibilidade de monitorar o *Eventlog,* que é formado por um conjunto de arquivos de *log* do sistema. Assim pode-se ter de forma mais ágil e centralizada qualquer informação sobre segurança e eventos do sistema, entre outras informações sobre as máquinas monitoradas. As figuras 19 e 20 exemplificam esse monitoramento.

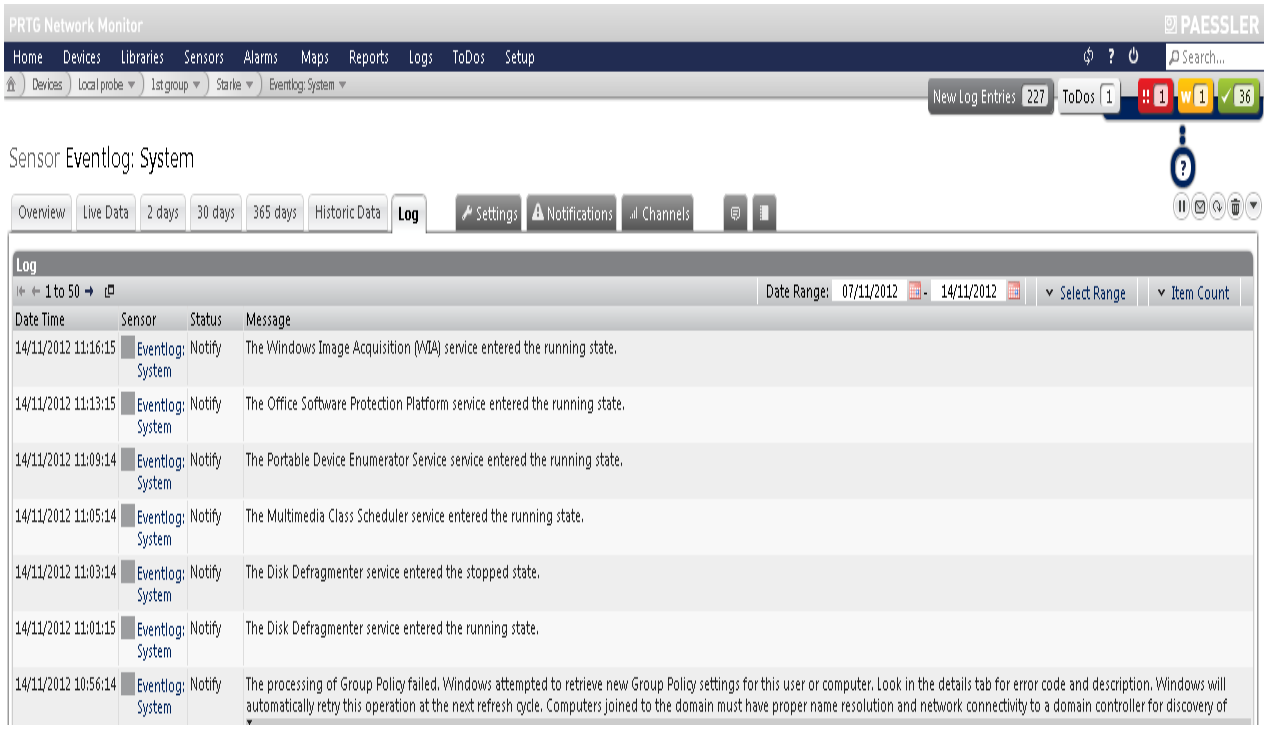



Figura 19 – Monitoramento dos eventos do sistema de uma máquina com o PRTG.

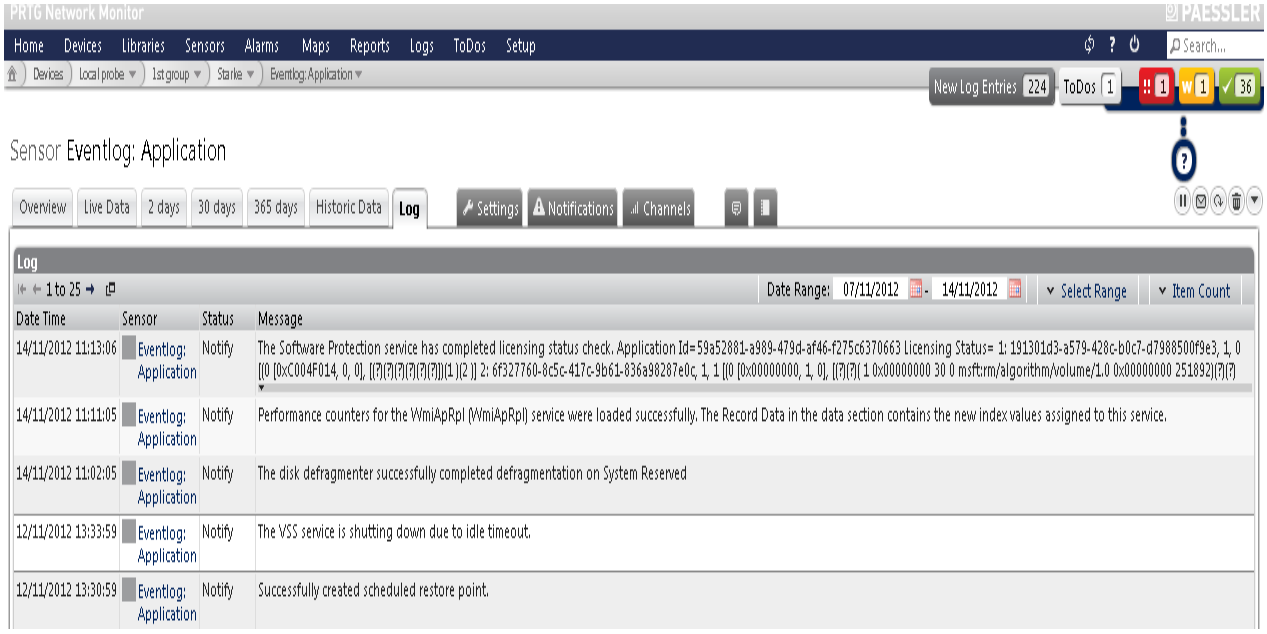



Figura 20 – Monitoramento dos eventos das aplicações de uma máquina com o PRTG.

### 4.2.3 **Monitoramento do Estado da Rede**

Como esse monitoramento tem-se o objetivo de avaliar do desempenho da rede, como por exemplo, o acesso à internet. Esse monitoramento também será feito com o uso do PRTG, pois além de seus gráficos serem muito informativos e de fácil compreensão, existe uma opção chamada *Ping Jitter* que possibilita medir a latência de trechos da rede.

No exemplo da figura 21 foi medida a variação do atraso de três trechos. Cada trecho representa a conexão até um roteador, sendo assim, será realizada a medição da latência até cada interface desse roteador, que é o próximo salto até realizar a comunicação com a *internet*. Sendo assim serão realizadas seis checagens, o que facilita a identificação de qual trecho pode estar gerando lentidão na comunicação. O primeiro ponto é até o *gateway* principal, o segundo até o próximo salto (*gateway 1*) e o terceiro até a o *Internet Service Provider* (ISP) (*gateway 2*). A figura 22 ilusta essa topologia.


Figura 21 – Trechos monitorados com o PRTG.

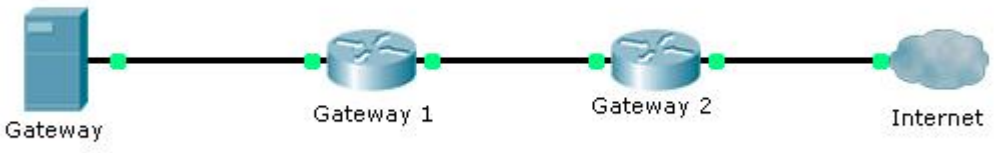



Figura 22 – Topologia da rede monitorada

Analisando a figura 23, pode-se perceber que o terceiro trecho que representa o ISP sofreu algum tipo de evento que gerou um jitter muito maior que a média. Caso essa variação houvesse ocorrido durante um longo período, fatalmente teria sido observada uma lentidão na comunica com a internet. Nessa mesma figura pode-se analisar que a latência da rede interna, mostrada no trecho 1, não sofreu nenhum tipo de variação, indicando que não há nenhum evento causando sobrecarga na rede.


Figura 23 – Gráfico da variação do jitter dos trechos monitorados com o PRTG.

Com os monitoramento descritos é possível ser ter informações relevantes sobre possíveis ocorrências. Assim possibilitando uma identificação mais fácil de algum evento, seja de uma violação de segurança ou do mal funcionamento de um equipamento.

# 5 **LICENCIAMENTO DOS SOFTWARES**

A maioria dos softwares utilizados nesse trabalho são livres, o que quer dizer que podem ser utilizados sem nenhum tipo de pagamento para obtenção de sua licença. Esse softwares são:

- *Squid* descrito na seção 3.1.2
- *DansGuardian* descrito na seção 3.1.2
- *Clamav* descrito na seção 3.1.2
- *Dovecot* descrito na seção 3.3
- *Postfix* descrito na seção 3.3
- *Mysql* descrito na seção 3.3
- *Apache* descrito na seção 3.3
- PHP descrito na seção 3.3
- *Logcheck* descrito na seção 4.1.3
- *Portaudit* descrito na seção 4.1.3
- *Nagios* descrito na seção 4.2.1
- *Cacti* descrito na seção 4.2.2

Tais softwares estão disponíveis no repositório do próprio sistema *FreeBSD* que foi usado nos exemplos e também é um sistema livre.

Já os softwares utilizados no sistema operacional Windows como:

- *SSH Secure Shell* descrito na seção 3.2.2
- *Snort* descrito na seção *4.1.1*
- *Mib Browser* descrito na seção *4.1.2*
- PRTG descrito na seção 4.2.3

Esses softwares também são livres, apesar dos dois últimos terem versões mais completas, porém pagas.

Então conforme visto é possível se ter uma boa estrutura de serviços, com a utilização de softwares conhecidos e reconhecidos, sem um grande investimento em licenciamento.

# 6 **CONCLUSÃO**

Conforme abordado nesse trabalho pode-se concluir que para manter uma rede e seus serviços seguros e funcionais é necessário muito trabalho e estudo. Cada sistema e aplicação tem suas especificidades, então para manter essa estrutura funcional deve ser feito um trabalho contínuo de monitoramento e atualização.

O uso de sistemas e aplicações reconhecidamente seguros e em constante atualização é importante na batalha contra os atacantes. E para obter informações sobre o estado dos sistemas e aplicações é preciso monitorar a infraestrutura.

Para facilitar a tarefa de monitorar os sistemas e aplicativos é necessário o uso de *softwares* que auxiliem essa função, a fim de obter uma solução mais eficiente de qualquer evento.

Então, com o uso de sistemas confiáveis, aplicativos para auxiliar na descoberta de vulnerabilidades e softwares de monitoramento, é possível manter a segurança de uma rede de uma forma mais eficaz.

# 7 **REFERÊNCIAS**

[1] Kurose, J. F e Ross, K. W Redes de Computadores e Internet: Uma abordagem top-down. 5ª ed. 2010

[2] Tanembaum, A. S. Computer Networks. 4ª ed. 2002

[3] Lipson, H. F. Tracking and Tracing Cyber-Attacks: Technical Challenges and Global Policy Issues, Pittsburgh, 2002, Carnegie Mellon University

[4] http://free.bsd.com.br/~eksffa/freebsd/ipfw.txt acessado em 12-09-2012

[5] http://www.hardware.com.br/livros/servidores-linux/usando-dansguardian.html acessado em 19-09-2012

[6] Rehman R. U. Intrusion Detection Systems with Snort : Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID 2003

[7] Improving Web Application Security: Threats and Countermeasures www.msdn.microsoft.com/en-us/library/ff648664.aspx acessado em 01-10-2012

[8] www.snort.com.br/comofuncionaids.asp acessado 19-09-2012

[9] http://www.cuca.in/sem-categoria/imap-x-pop-qual-escolher/ acessado 19-09-2012

[10] http://news.netcraft.com/ acessado em 19-09-2012

[11] http://zone-h.org/archive acessado em 19-09-2012

[12] Gite, V. Linux: 25 PHP Security Best Practies for Sys Admins 2011

[13]http://searchfinancialsecurity.techtarget.com/news/1293647/How-to-protect-and-harden-a-database-server acessado 03-10-2012

[14] http://www.applicure.com/blog/database-security-best-practice acessado 03-10-2012

[15] http://www.hardened-php.net/suhosin/index.html acessado 02-10-2012

[16]http://www.cisco.com/en/US/docs/switches/lan/catalyst4500/12.1/19ew/configuration/guide/port_sec.pdf acessado 27-09-2012

[17] http://sourceforge.net/apps/mediawiki/mod-security/index.php?title=Reference_Manual acessado 27-09-2012

[18] http://www.aisecure.net/2011/09/24/securing-web-application-servers/ acessado 02-10-2012

[19] http://www.symantec.com/connect/articles/web-security-appliance-apache-and-modsecurity acessado 02-10-2012

[20] http://oreilly.com/catalog/dns4/chapter/ch11.html acessado 04-10-2012

[21] http://www.net.cmu.edu/groups/netdev/docs/bind9/Bv9ARM.ch06.html acessado 04-10-2012

[22] http://www.cert.br/docs/whitepapers/dns-recursivo-aberto/#4 acessado 05-10-2012

[23] https://labs.nic.cz/files/labs/DNS-cache-poisoning-attack-analysis.pdf acessado 04-05-2012

[24] http://registro.br/suporte/faq/faq8.html acessado 05-10-2012

[25] ftp://ftp.registro.br/pub/doc/introducao-dns-dnssec.pdf acessado 05-10-2012

[26] ftp://ftp.registro.br/pub/doc/tutorial-dnssec.pdf acessado 06-10-2012

[27] ftp://ftp.registro.br/pub/doc/configuracao_dnssec_servidor_recursivo.pdf acessado 06-10-2012

[28] http://www.postfix.org/SASL_README.html acessado 09-10-2012

[29] http://www.postfix.org/TLS_README.html acessado 09-10-2012

[30] http://www.hardware.com.br/livros/linux-redes/autenticando-clientes.html acessado 10-10-2012

[31] http://www.hardware.com.br/livros/linux-redes/ativando-tls.html acessado 10-10-2012

[32] http://johnny.chadda.se/article/mail-server-howto-postfix-and-dovecot-with-mysql-and-tlssl-postgrey-and-dspam/ acessado 10-10-2012

[33] http://www.unitednerds.org/thefallen/docs/index.php?area=Postfix&tuto=Postfix-AntiSpam-1 acessado 11-10-2012

[34]http://www.postfix.org/postconf.5.html acessado 11-10-2012

[35] http://silverwraith.com/papers/freebsd-ddos.php acessado 15-10-2012

[36] http://etherealmind.com/tcp-syn-cookies-ddos-defence/ acessado 15-10-2012

[37] http://biosystems.ath.cx/wiki/doku.php?id=manuais:sar sessão Tunning de rede acessado 15-10-2012

[38 ] Mauro, D. R. e Schmidt, K. J. Essential SNMP 2ª ed. 2005

[39] http://www.freshports.org/security/logcheck/ acessado em 30-10-2012

[40] http://www.freebsd.org/doc/handbook/security-portaudit.html acessado 01-11-2012

[41] http://nagiosnapratica.wordpress.com/ acessado 05-11-2012

[42] http://www.nagios.org/ acessado 05-11-2012

[43] http://www.cacti.net/ acessado 08-11-2012

[44] http://www.paessler.com/prtg 09-11-2012